Semestrálne zadanie: Komunikácia s využitím UDP protokolu

by

Tomáš Meravý Murárik

at Faculty of Informatics and Information Technologies STU

Course: Computer and Communication Networks

Assignment object: Design and implement P2P(Peer to Peer) application using custom protocol built on top of UDP (User Datagram Protocol) in the transport layer of TCP/IP model. The application should allow 2 users to communication over local Ethernet network, including text transmission and exchange of files between computers (nodes) . Both nodes will work simultaneously as receiver and sender.

Table of Contents

Structure of protocol header

32 bits

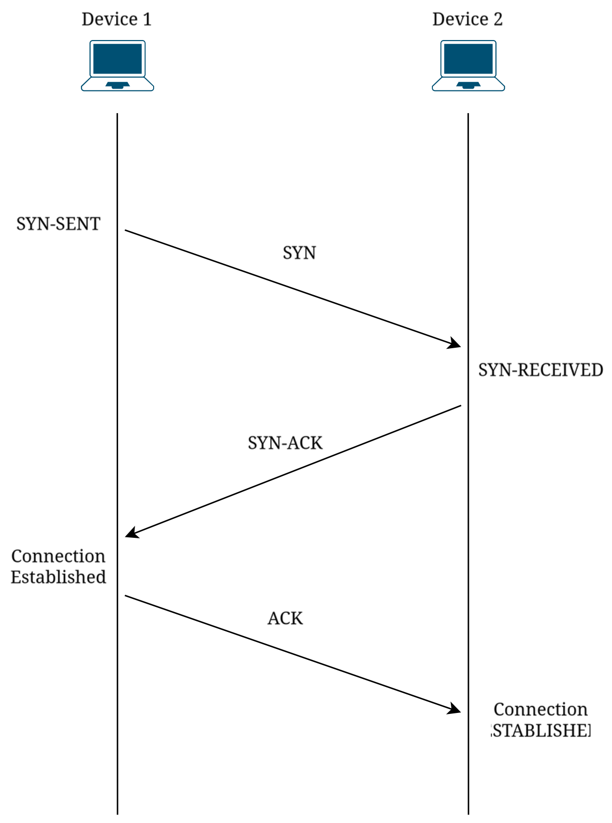
flag	Current fragment number	Total num of fragments
Total num of fragments	length	data 10-255
data 10-255	crc16	

Estabilishing connection:[1]

Similaraly to TCP I will be using 3 way handshake using SYN-ACK system.

1. SYN) Both clients will be sending SYN packets to specified ports till one of the responds with SYN-ACK.
2. SYN-ACK)After SYN packet is recievied they will send back SYN-ACK packet acknowledging that they received SYN packet and is waiting for his ACK.
3. ACK) After client gets SYN-ACK packet he will respond with ACK packet completing the 3 way handshake.

Protocol uses the Flags field to signal which control state it’s using.



1

Fragmentation:

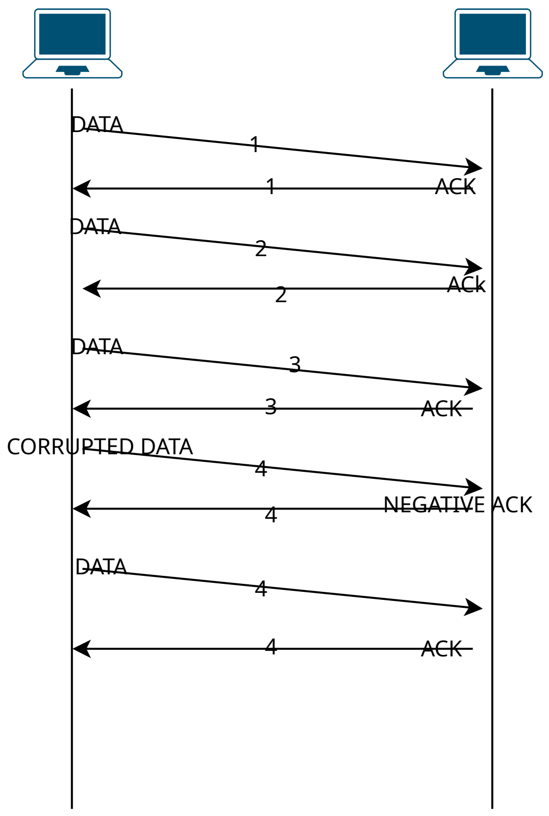
Based on sequence number and fragment number , the program will determine how many packets it should expect and how it will reassemble the packet back together once all packets have been received.

Error detection :

Using CRC16 the program will use checksum value to determine whether the received packet is corrupted or not.

CRC16:This algorithm uses 16 bit polynomial (divisor) to perform bitwise division on the data using binary xor operations where the remainder of there operation is appended at the end of the header . Same operation is performed by the receiver and if the receivers operation is equal to 0 , the client will return True saying that the data is intact . If the operation is not equal the algorithm will return False and the receiver will ask for the data again.

Reliable Data Transmission (ARQ):
After receiving each packet, the receiver sends an acknowledgment packet confirming successful reception. If the packet is corrupted, a negative acknowledgment (NACK) is sent, requesting retransmission. If a fragment is missing during larger data transfers, the receiver will ask for it using its sequence number, identifying the missing fragment.
Using Stop-and-Wait ARQ, the sender will wait for an acknowledgment (ACK) before sending the next packet. If the sender receives a NACK or times out waiting for an ACK, it will resend the packet until an ACK is received.



Keep Alive:

The keep-alive mechanism will periodically send messages (e.g., every 30 seconds) from the sender to the receiver to indicate that the connection should remain active. The receiver is expected to respond with an acknowledgment (ACK). If no ACK is received after several keep-alive attempts, the connection will be closed.

Data corruption simulation :

To check whether error detection and fragmentation works , the sender will have option to send bad packet on purpose by altering checksum so the error detection on the receiving end asks for the packet again.
Connection termination :
Connection will be terminated in a similar way to tcp where the sender will announce the connection termination with a FIN flag and will wait for receiver to respond with ack .

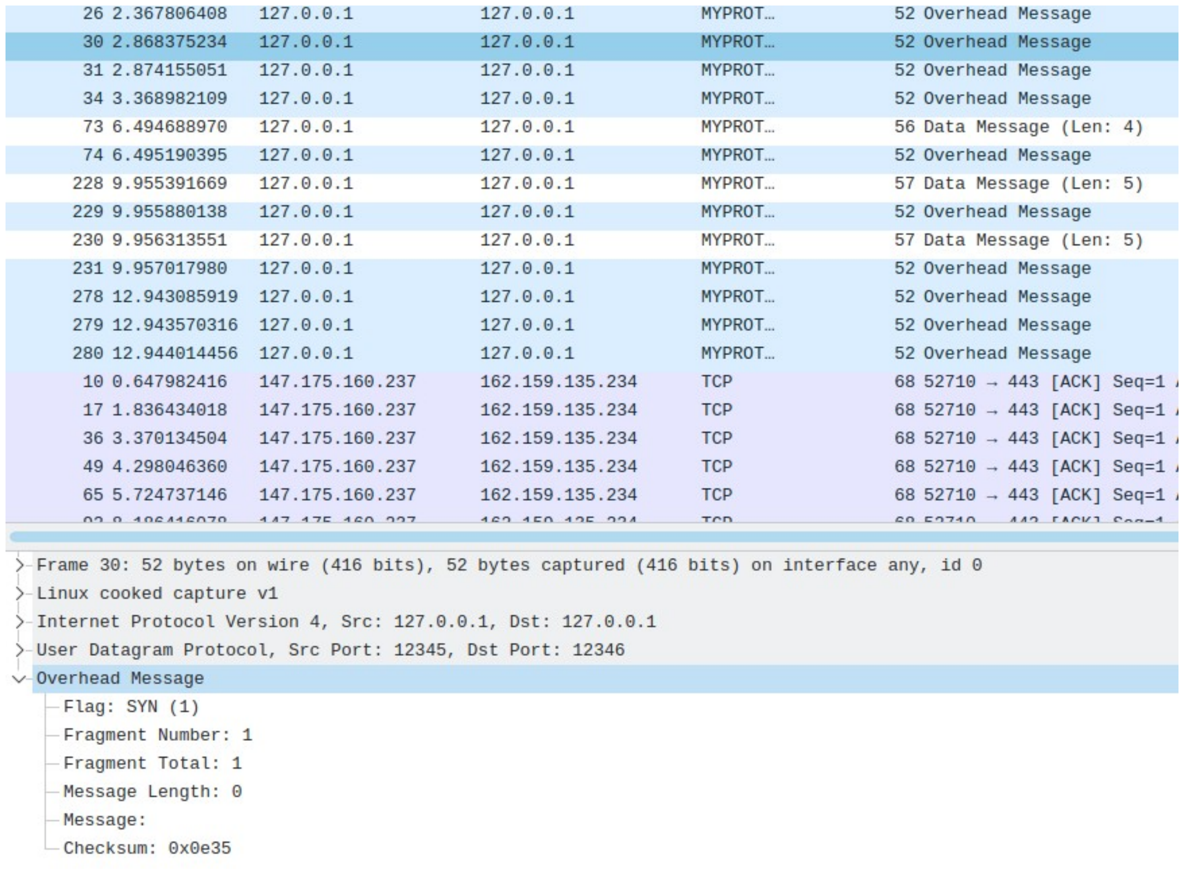
Changes made during implementation :

2 changes made during the making of this program were in error detection and in header structure.

- Error detection) Instead of comparing crc16 value of message except last 2 bytes on receivers end with crc16 of sender (last 2 bytes) , I decided to compare crc16 of whole message to 0 which should be clear indicator if message is corrupted or not.
- Header length) In my previous protocol version I wanted to have header length of 1byte but after careful consideration I decided to make it into 2 bytes.

Example comunication:

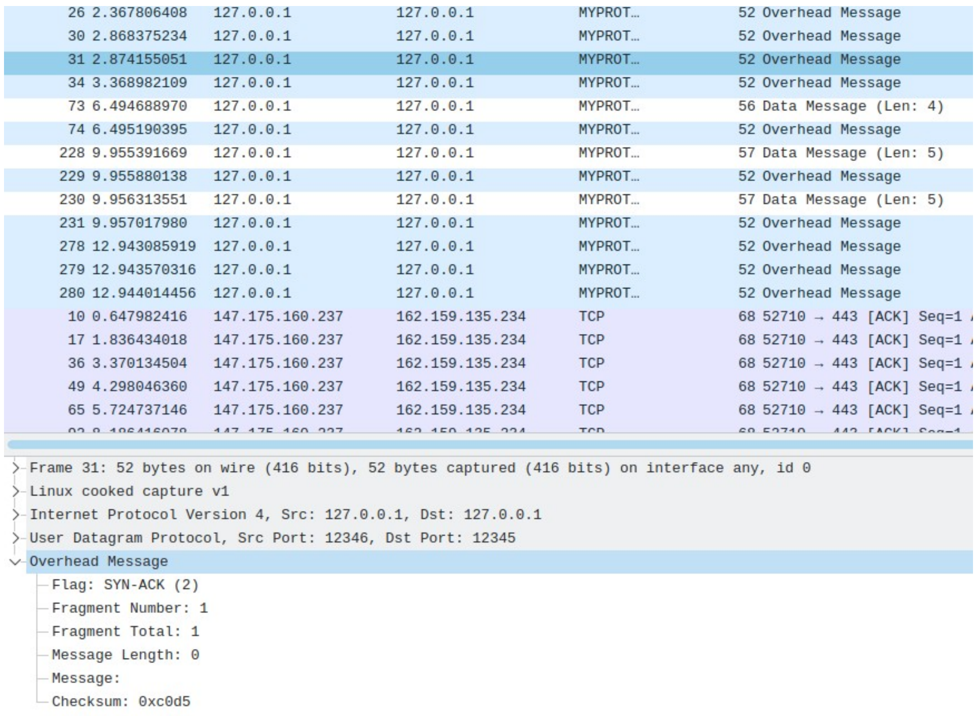
Handshake : SYN-packet announcing that the client1 is ready to start communication



```
 26 2.367806408   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
 30 2.868375234   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
 31 2.874155051   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
 34 3.368982109   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
 73 6.494688970   127.0.0.1         127.0.0.1         MYPROT…        56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
228 9.955391669   127.0.0.1         127.0.0.1         MYPROT…        57 Data Message (Len: 5)
229 9.955880138   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
230 9.956313551   127.0.0.1         127.0.0.1         MYPROT…        57 Data Message (Len: 5)
231 9.957017980   127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
278 12.943085919  127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
279 12.943570316  127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
280 12.944014456  127.0.0.1         127.0.0.1         MYPROT…        52 Overhead Message
 10 0.647982416   147.175.160.237   162.159.135.234   TCP            68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237   162.159.135.234   TCP            68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237   162.159.135.234   TCP            68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237   162.159.135.234   TCP            68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237   162.159.135.234   TCP            68 52710 → 443 [ACK] Seq=1

> Frame 30: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12345, Dst Port: 12346
v Overhead Message
    Flag: SYN (1)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x0e35
```

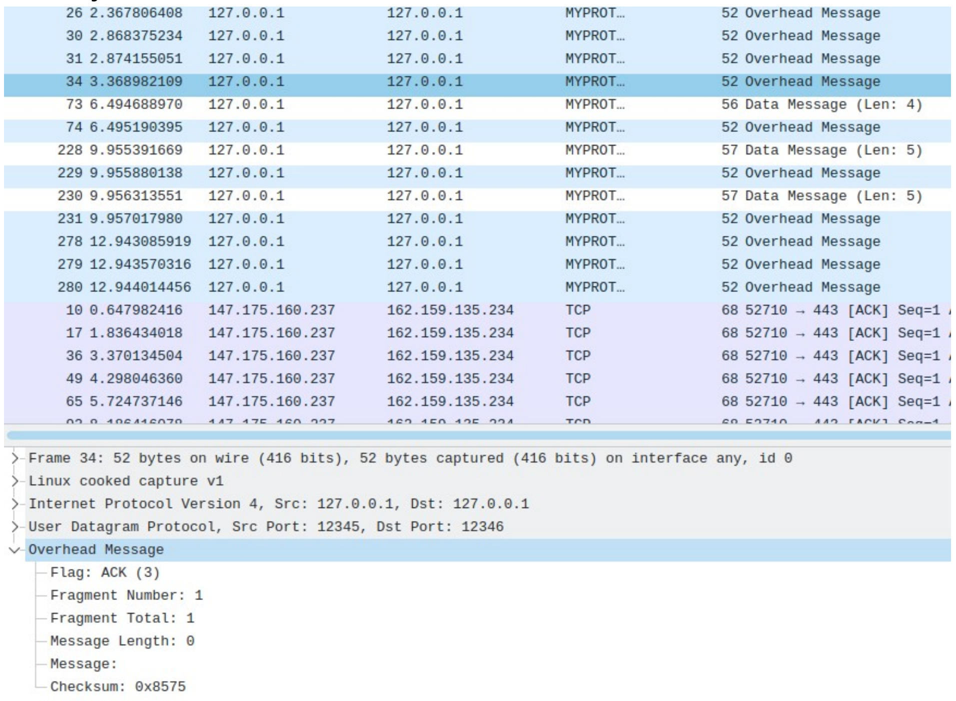
SYN-ACK-packet telling the client2 is also ready to start communication



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 31: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12346, Dst Port: 12345
v Overhead Message
    Flag: SYN-ACK (2)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0xc0d5
```

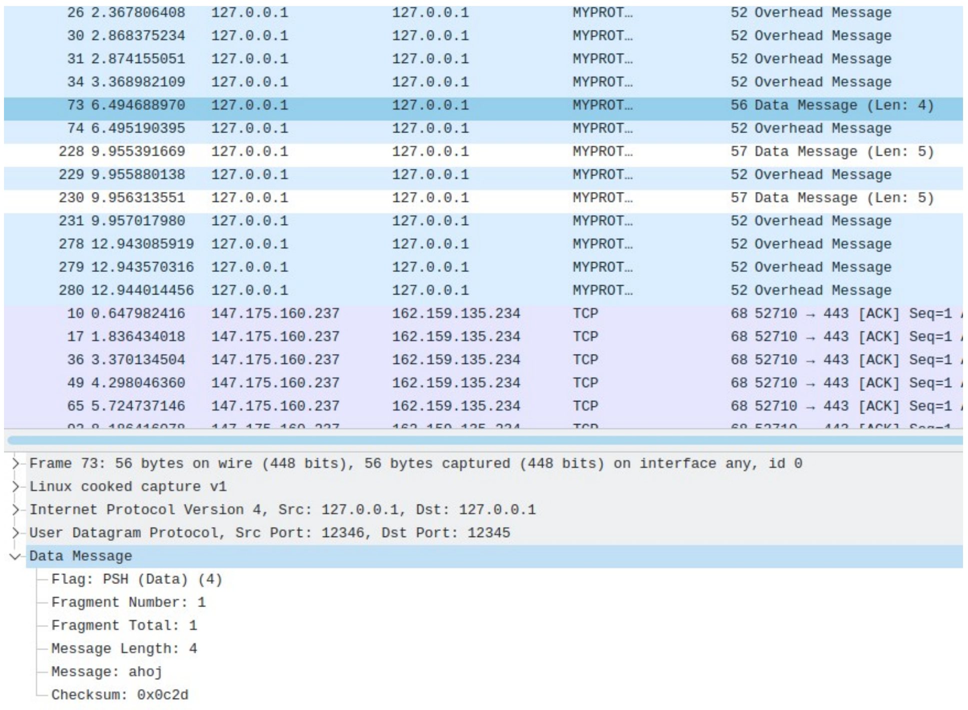
ACK-packet from client1 telling client2 that they received their syn-ack packet successfully



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 34: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12345, Dst Port: 12346
v Overhead Message
    Flag: ACK (3)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x8575
```

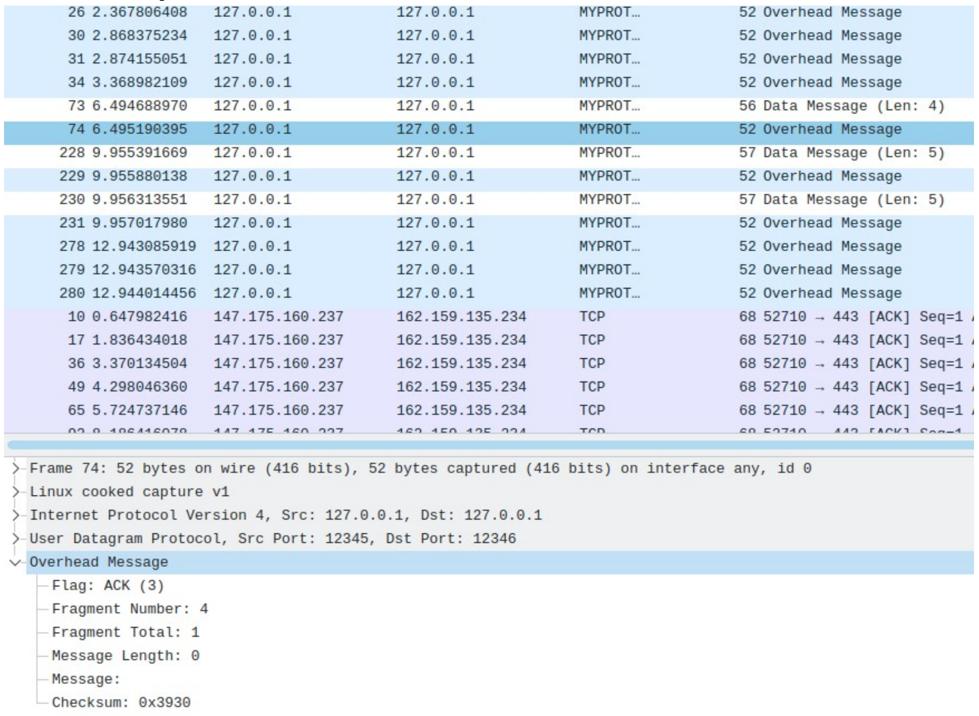
Sending not corrupted data:
PSH-4 :client2 (port:12346) is sending data to client1 and is immediately starting to listen for response



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 73: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12346, Dst Port: 12345
v Data Message
    Flag: PSH (Data) (4)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 4
    Message: ahoj
    Checksum: 0x0c2d
```

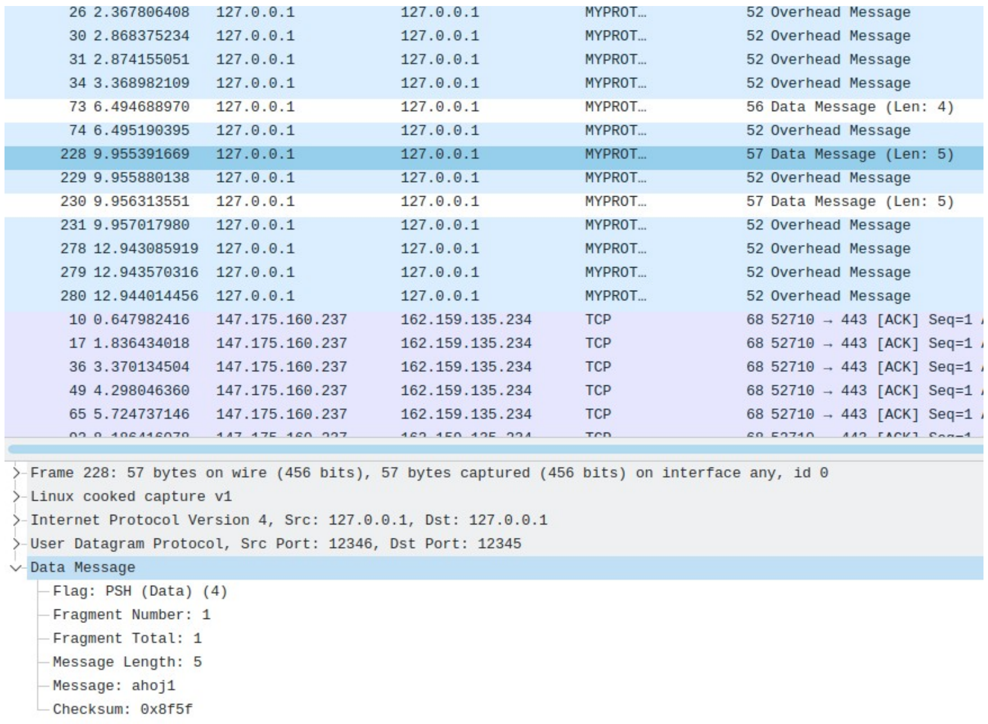
ACK: client1 successfully receibed the data and is telling the other client that everything went correctly



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 74: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12345, Dst Port: 12346
v Overhead Message
    Flag: ACK (3)
    Fragment Number: 4
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x3930
```

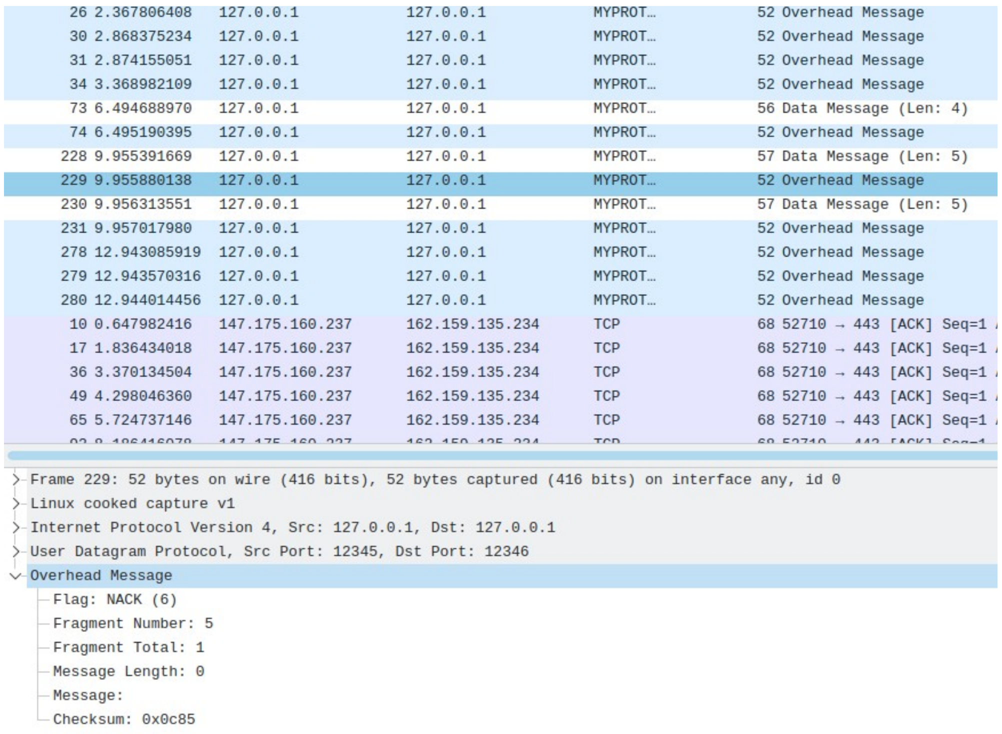
Sending corrupted data:
PSH-4 :client2 (port:12346) is sending corrupted data to client1 by adding +1 to checksum



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
```

```
> Frame 228: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12346, Dst Port: 12345
v Data Message
    Flag: PSH (Data) (4)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 5
    Message: ahoj1
    Checksum: 0x8f5f
```

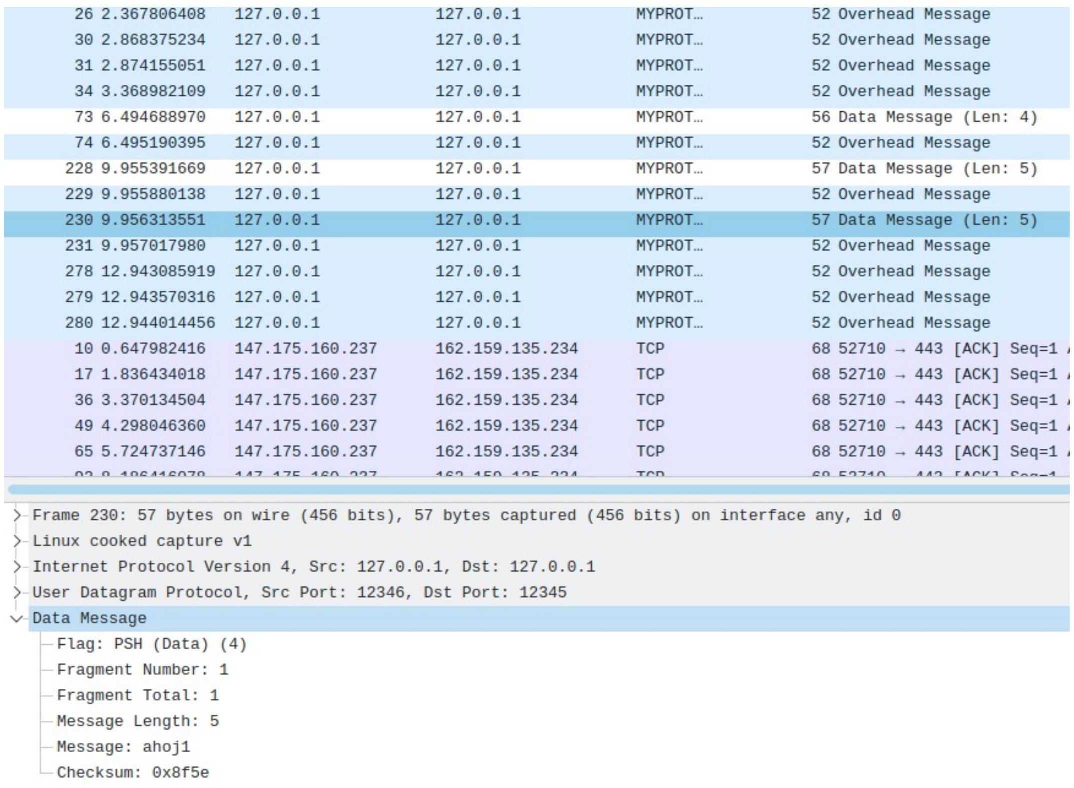
NACK : on receival of corrupted data the client 1 will respond with NACK message , telling the sending client that something went wrong.



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
```

```
> Frame 229: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12345, Dst Port: 12346
v Overhead Message
    Flag: NACK (6)
    Fragment Number: 5
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x0c85
```

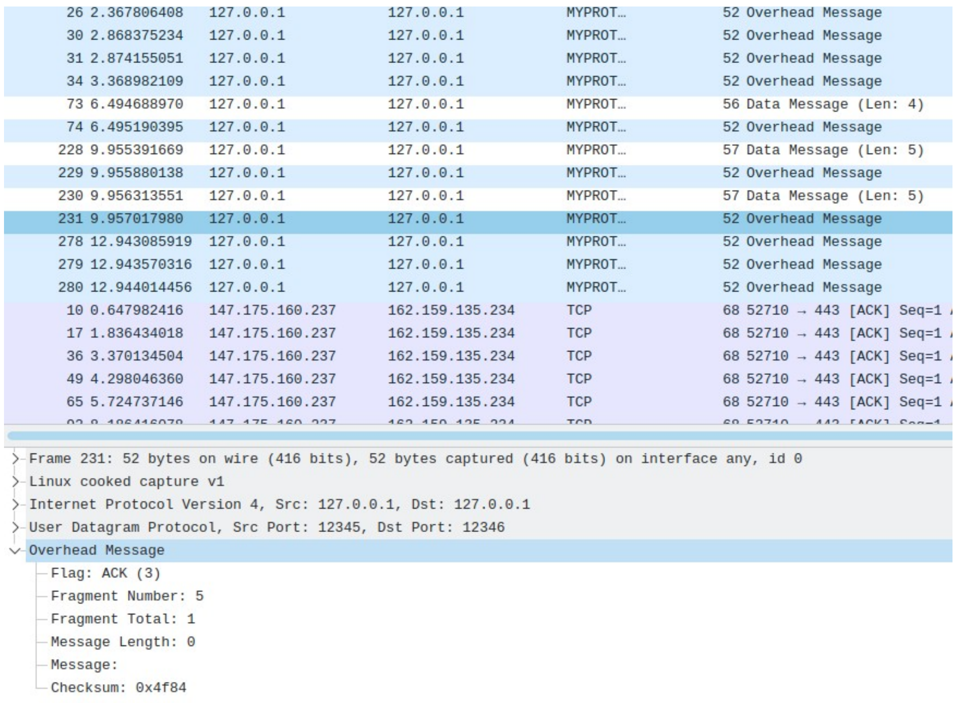
Resending data:
PSH :upon receival of NACK packet , the sending client sends the data again.



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 230: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12346, Dst Port: 12345
v Data Message
    Flag: PSH (Data) (4)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 5
    Message: ahoj1
    Checksum: 0x8f5e
```

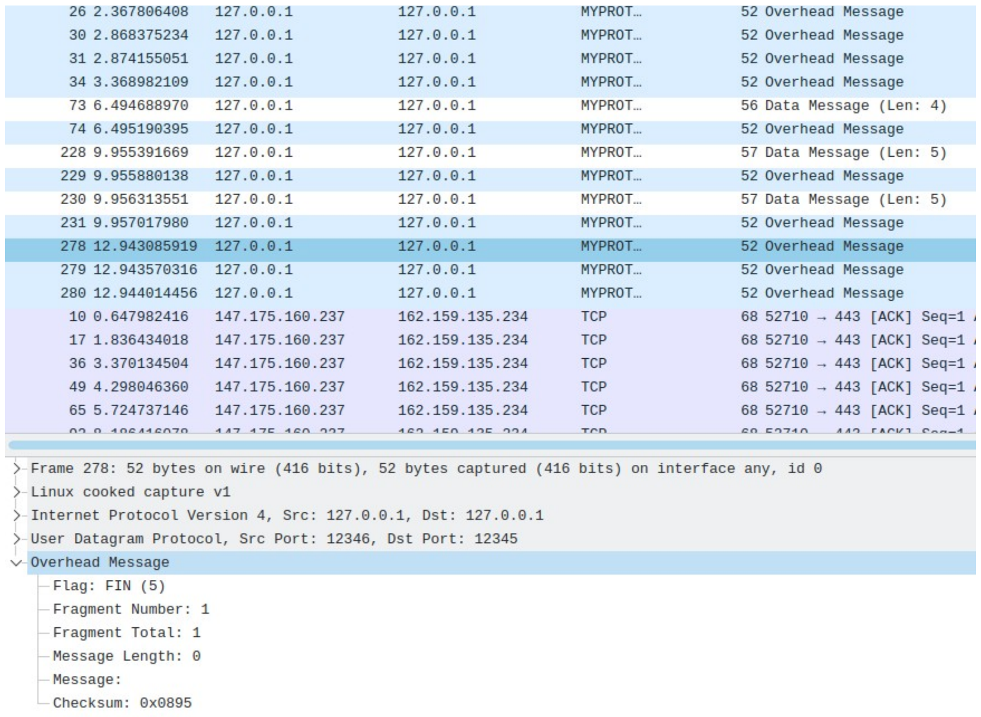
ACK : When the receiving client finally receives correct data, he sends ack packet.



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 231: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12345, Dst Port: 12346
v Overhead Message
    Flag: ACK (3)
    Fragment Number: 5
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x4f84
```

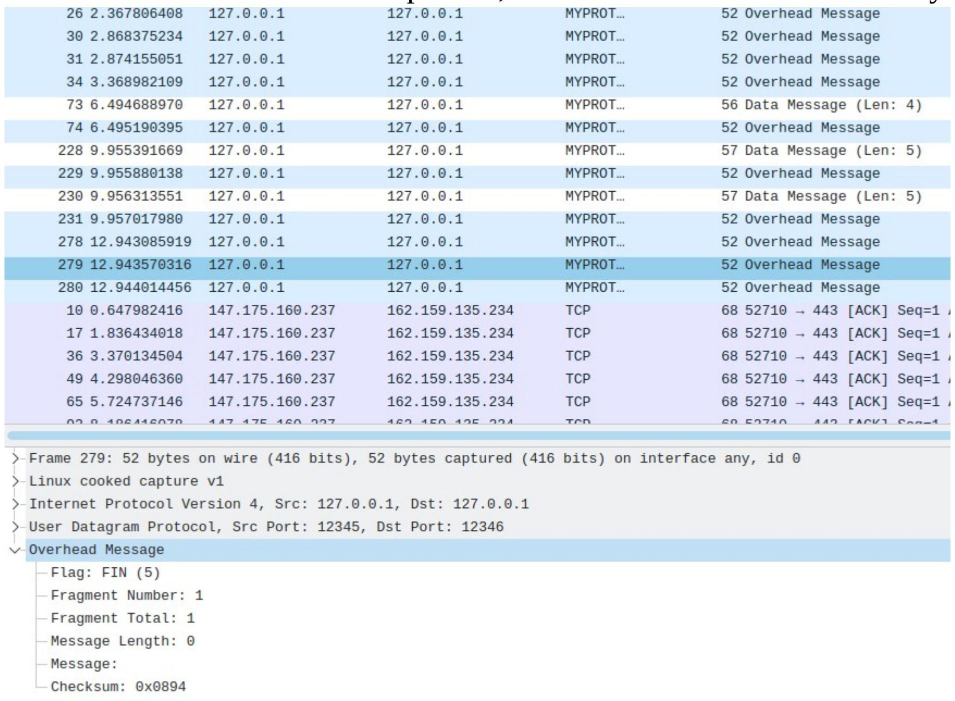
Ending connection:
FIN-If client one of the clients decides to end the conversation , he sends a FIN-packet telling the other end that he intends to stop the communication.



```
 26 2.367806408    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 30 2.868375234    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 31 2.874155051    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 34 3.368982109    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 73 6.494688970    127.0.0.1        127.0.0.1        MYPROT…     56 Data Message (Len: 4)
 74 6.495190395    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
228 9.955391669    127.0.0.1        127.0.0.1        MYPROT…     57 Data Message (Len: 5)
229 9.955880138    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
230 9.956313551    127.0.0.1        127.0.0.1        MYPROT…     57 Data Message (Len: 5)
231 9.957017980    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
278 12.943085919   127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
279 12.943570316   127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
280 12.944014456   127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 10 0.647982416    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 17 1.836434018    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 36 3.370134504    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 49 4.298046360    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 65 5.724737146    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1

> Frame 278: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12346, Dst Port: 12345
v Overhead Message
    Flag: FIN (5)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x0895
```

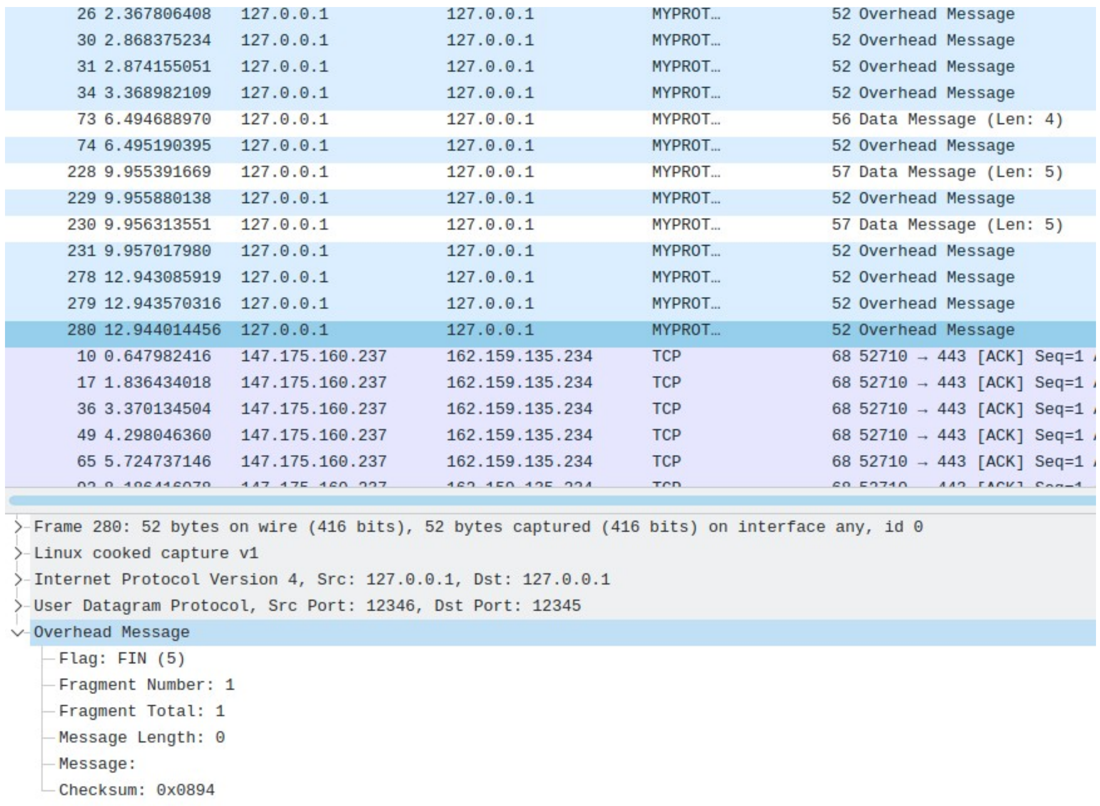
FIN : When client receives FIN packet , he sends one back and immedialy closes itself.



```
 26 2.367806408    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 30 2.868375234    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 31 2.874155051    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 34 3.368982109    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 73 6.494688970    127.0.0.1        127.0.0.1        MYPROT…     56 Data Message (Len: 4)
 74 6.495190395    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
228 9.955391669    127.0.0.1        127.0.0.1        MYPROT…     57 Data Message (Len: 5)
229 9.955880138    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
230 9.956313551    127.0.0.1        127.0.0.1        MYPROT…     57 Data Message (Len: 5)
231 9.957017980    127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
278 12.943085919   127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
279 12.943570316   127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
280 12.944014456   127.0.0.1        127.0.0.1        MYPROT…     52 Overhead Message
 10 0.647982416    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 17 1.836434018    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 36 3.370134504    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 49 4.298046360    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1
 65 5.724737146    147.175.160.237  162.159.135.234  TCP         68 52710 → 443 [ACK] Seq=1

> Frame 279: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12345, Dst Port: 12346
v Overhead Message
    Flag: FIN (5)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x0894
```

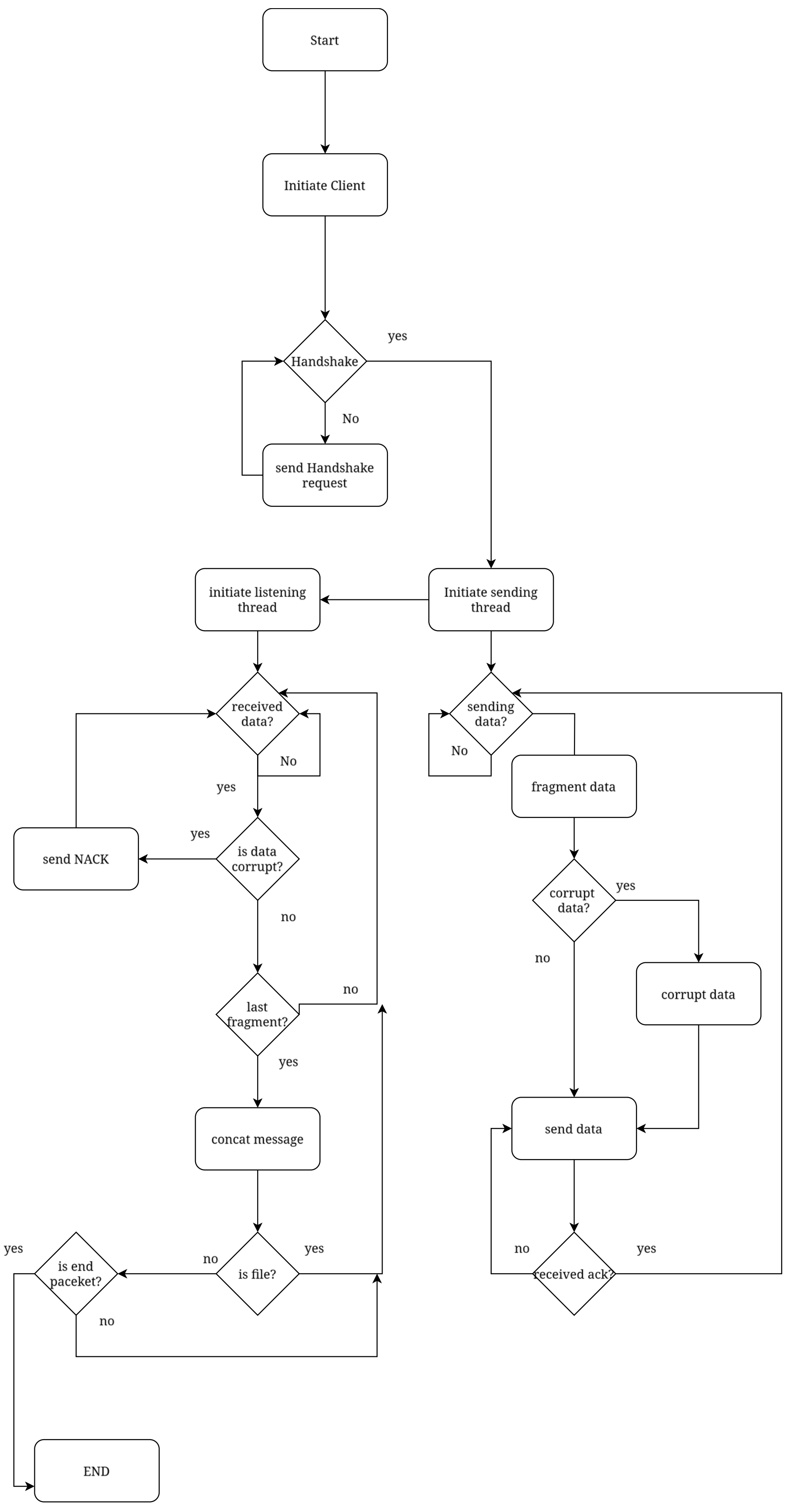
FIN:finally when the first client receives FIN , he closes itself (completing 3 way finishig handshake)



```
 26 2.367806408   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 30 2.868375234   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 31 2.874155051   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 34 3.368982109   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 73 6.494688970   127.0.0.1        127.0.0.1        MYPROT…   56 Data Message (Len: 4)
 74 6.495190395   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
228 9.955391669   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
229 9.955880138   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
230 9.956313551   127.0.0.1        127.0.0.1        MYPROT…   57 Data Message (Len: 5)
231 9.957017980   127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
278 12.943085919  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
279 12.943570316  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
280 12.944014456  127.0.0.1        127.0.0.1        MYPROT…   52 Overhead Message
 10 0.647982416   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 17 1.836434018   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 36 3.370134504   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 49 4.298046360   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1
 65 5.724737146   147.175.160.237  162.159.135.234  TCP       68 52710 → 443 [ACK] Seq=1

> Frame 280: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface any, id 0
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
> User Datagram Protocol, Src Port: 12346, Dst Port: 12345
v Overhead Message
    Flag: FIN (5)
    Fragment Number: 1
    Fragment Total: 1
    Message Length: 0
    Message:
    Checksum: 0x0894
```

Program diagram:



Conclusion:

In conclusion, this project successfully implements a custom peer-to-peer communication protocol using UDP, ensuring reliable data transmission through features like a three-way handshake, CRC16 error detection, and fragmentation. This project all the basic requirements and successfully implements all mandatory conditions.

Bibliography

1: Nathan Jennings, Socket Programming in Python (Guide), 2022